목차

1. Lambda

1-1. API GateWay - Lambda의 연결

1-2. Lambda - Dynamo DB의 연결

1-2-1. 엑세스 에러

1-2-2. 해결방안 : IAM에서 role, policy 추가

1. 전체 동작
2. Test에서의 확장

3-1. Get 메소드

3-1-1. 테이블 전체 리턴대신, 한 (특정) 항목만 리턴받기

-------------------------------------------

3-1-2. json으로 데이터 추가 7월 15일 학습내용으로 이월

3-1-3. json으로 데이터 변경

3-1-4. json으로 데이터 삭제

----------------------------------------------

1. 웹 뷰 어플리케이션 제작

1. Lambda

: 결과를 내보내기 위해서는 Handler() 함수를 사용해야한다.

따라서

Export 객체의 메소드로

export.handler async (param1 , param2 ) => {

let body;

// body = await dynamo.scan(param1).promise(); //dynamo db 실행내용

return body;

}

으로 수행하여야 한다.

* async (비동기) 를 뺄 경우 원하는 값이 제대로 나오지 않을 수도 있다.
* let 은 var와 다를게 없지만, 차이점은 재정의가 가능하다는 것 이다.
* 이렇게 쓴다면 param1에는 클라이언트로 전달 받은 내용이,
* param2에는 람다의 정보가 들어가게 된다.

Lambda에서 취할수 있는 dynamo db의 실행 내용으로는 크게

// body = await dynamo.scan(param1).promise();

// body = await dynamo.put(param1).promise();

// body = await dynamo.update(param1).promise();

// body = await dynamo.delete(param1).promise();

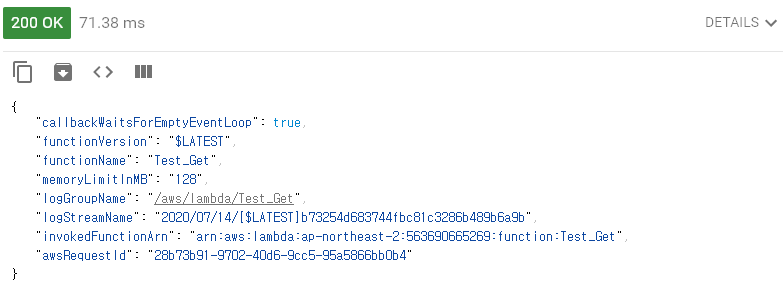
가 있다.

1-1. API GateWay - Lambda의 연결



Deploy 주소를 이용하여 아래의 주소로 요청을 보내본 결과

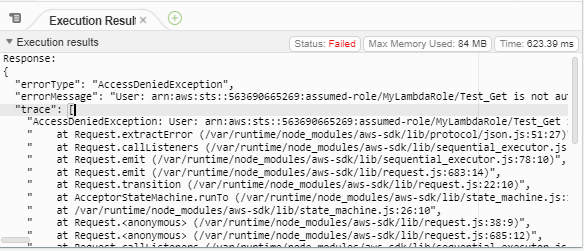
https://e5sggq8tjg.execute-api.ap-northeast-2.amazonaws.com/TestDeploy/Test\_Get



제대로 응답을 받았음을 확인하였다. (연결 완료)

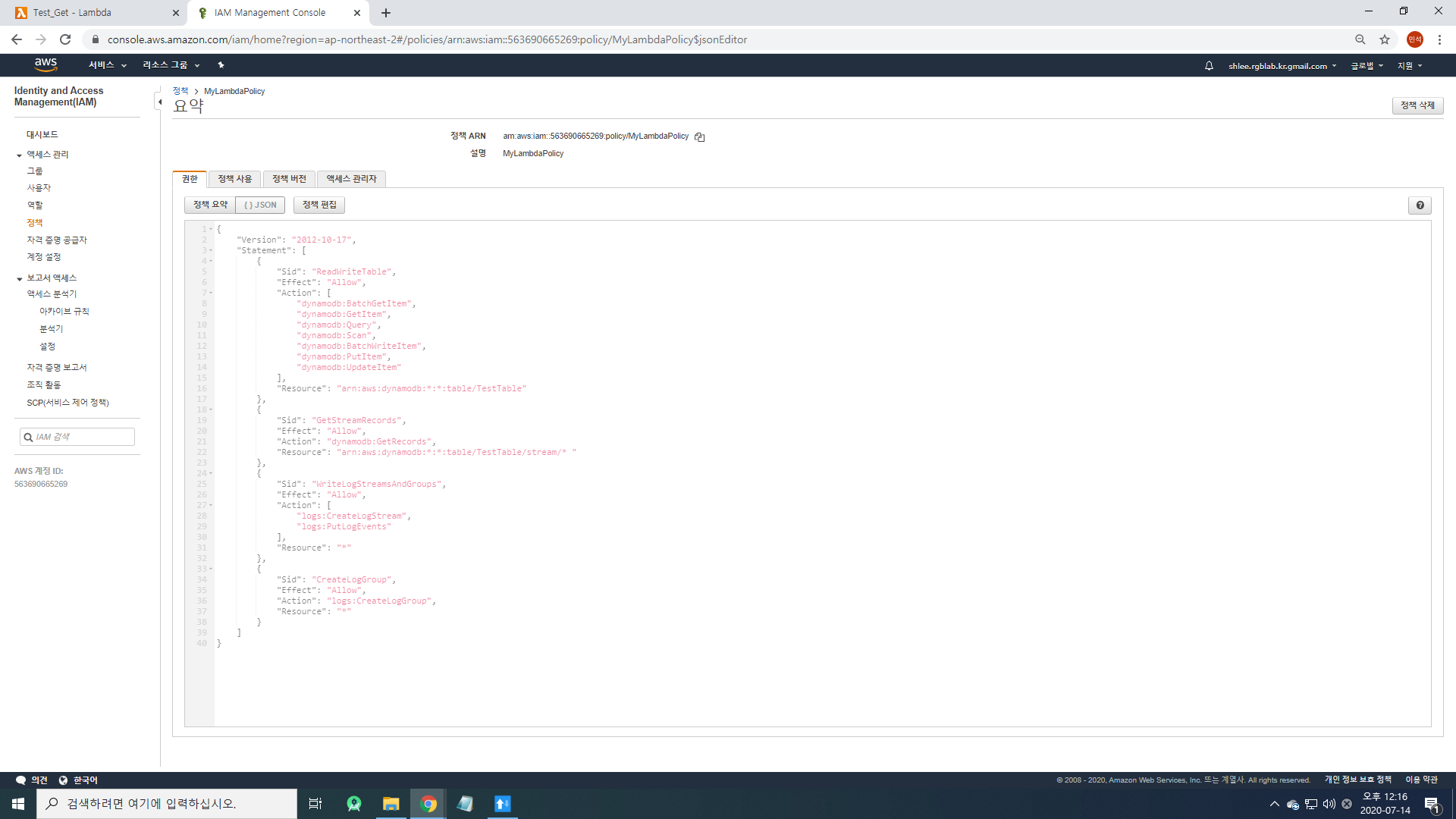
**1-2. Lambda - Dynamo DB의 연결**

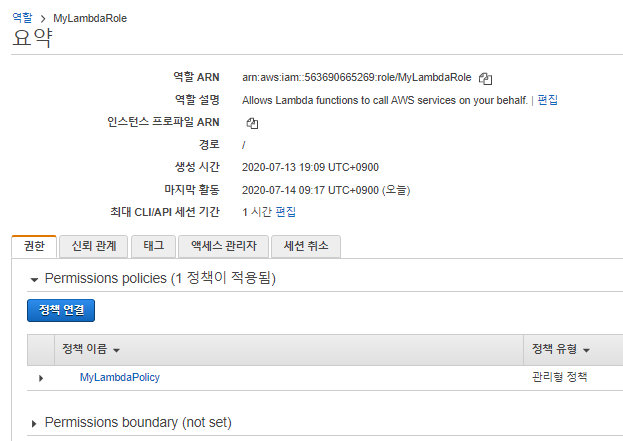
**1-2-1. 엑세스 에러**

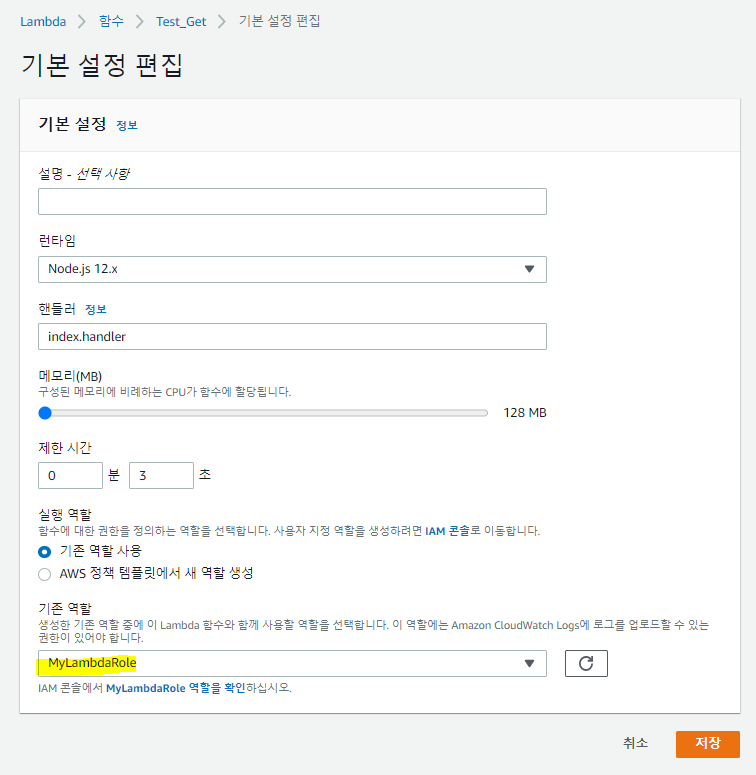
람다에서 테스트를 돌려본 결과 다음과 같은 결과를 받았다.

즉, access가 거부되어 있는 상태이다.

**1-2-1. 해결 방안**







다이나모 DB 에 관한 정책을 만든 후

이를 역할에 넣어주고 나니 Access Denied는 해결되었다.

아래의 json으로 다시 테스트 해보니

{

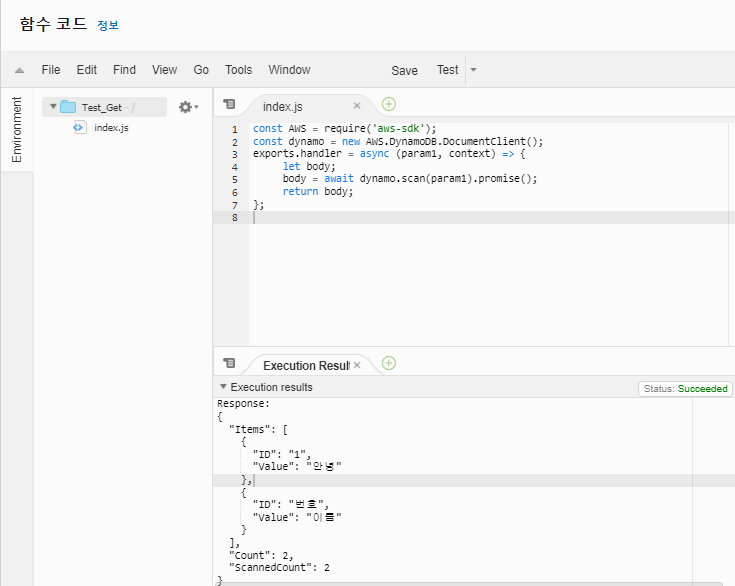
"TableName": "TestTable"

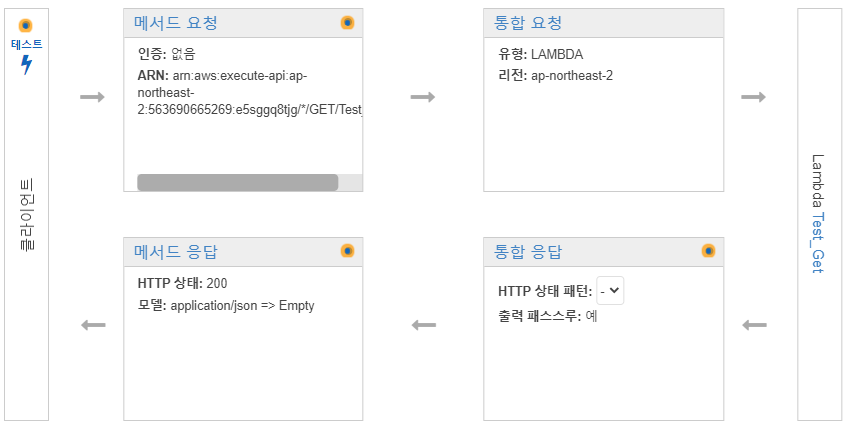
}

성공적으로 결과가 나왔다.

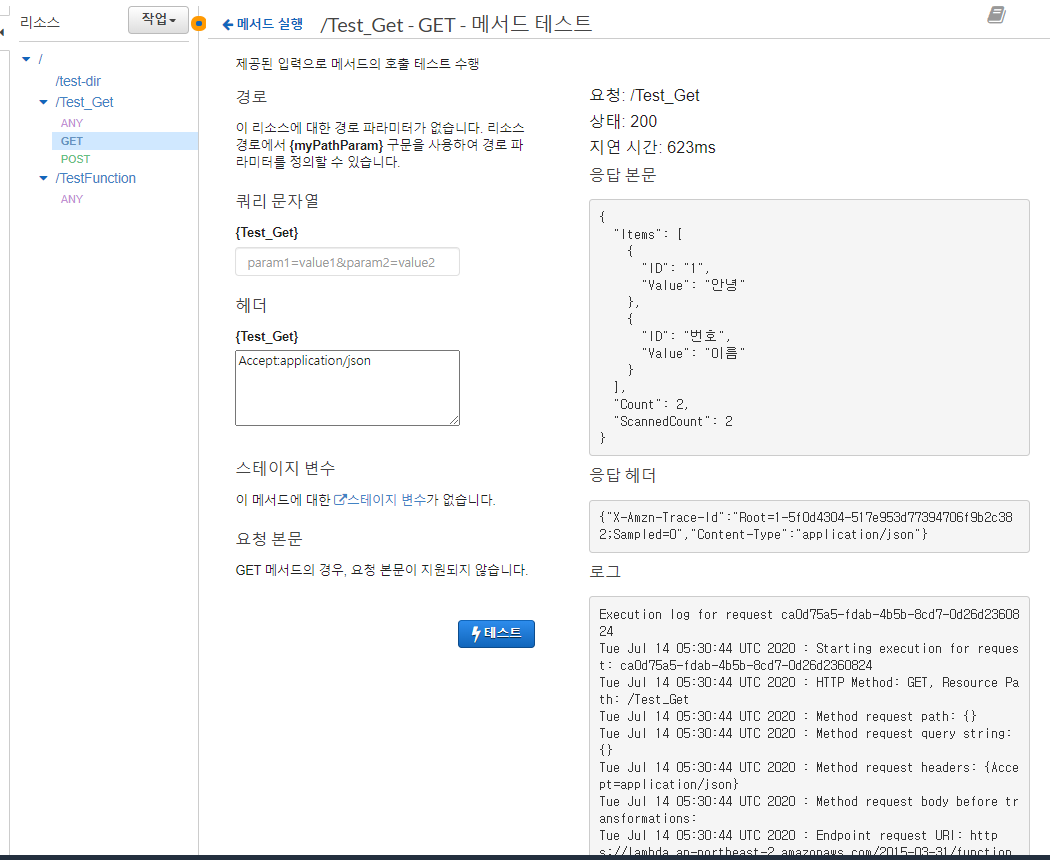
**2. 전체 동작**

이걸 이제 쭉 연결하여 테스트해보면 아래와 같이 성공적으로 요청을 보내고 응답을 받았다.





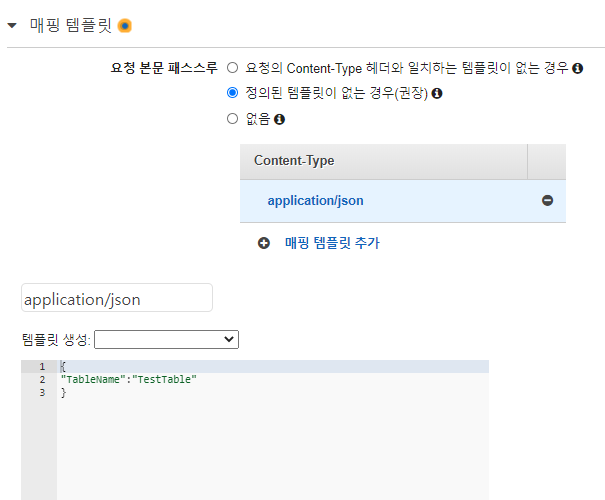
좌측 테스트를 클릭 하면 다음의 화면이 나온다.



여기서 클라이언트에서 추후 json 형식으로 보내온다고 가정하여

application/json을 사용하여 get 요청을 보내보았다.

application/json은 ‘통합 요청’ - ‘매핑 템플릿’에 정의되어 있다.



3. Test에서의 확장

3-1. Get 메소드

3-1-1. 테이블 전체 리턴대신, 한 (특정) 항목만 리턴받기

----------------------------------------------------------------------------------------------------------------

1. ProjectionExpression : DB에서 배웠던 projection 하는 부분(이 attribute들만 뽑아내겠다.)
2. FIlterExpression : 원하지 않는 것들을 제거 할 수 있다.

=> 이를 적용시켜 ID와 Value만 Projection 해보고 FilterExpression으로 Filter 항목이 true 인 것을 걸러보려고 한다.

현재 TestTable에는



이러한 내용이 들어가있다.

방법 1)

매핑 템플릿의 application/json -> 요청할 때의 json의 내용을 바꾸거나

|  |  |
| --- | --- |
| {  "TableName":"TestTable",  "ProjectionExpression" :"ID"  } | const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  let body;  body = await dynamo.scan(param1).promise();  return body;  }; |

단, Table에서 스캔으로 되어있는 것에 대해서만 ProjectionExpression

가능 한 것으로 보인다.

|  |  |
| --- | --- |
|  |  |

방법 2)

람다 함수에서 처리

|  |  |
| --- | --- |
| const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  var param = {  TableName : "TestTable",  KeyConditionExpression: "#id = :y",  ExpressionAttributeNames:{  "#id": "ID" },  ExpressionAttributeValues: {  ":y": "1" }  };  exports.handler = async (param1, context) => {  let body;  body = await dynamo.query(param).promise();  return body; }; |  |

오늘 다 못한 것

3-1-2. json으로 데이터 추가

3-1-3. json으로 데이터 변경

3-1-4. json으로 데이터 삭제

내일 해야 하는 것

POST 방식으로 테이블 접근

======================================================================

**4. Web View 어플리케이션 제작**

package com.example.webview;

import android.os.Build;

import android.os.Bundle;

import android.view.View;

import android.view.Window;

import android.webkit.WebSettings;

import android.webkit.WebView;

import android.webkit.WebViewClient;

import androidx.appcompat.app.AppCompatActivity;

import com.example.webview.R;

public class MainActivity extends AppCompatActivity {

private WebView mWebView; // 웹뷰 선언

private WebSettings mWebSettings; //웹뷰세팅

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

// 웹뷰 시작

mWebView = (WebView) findViewById(R.id.*webView*);

mWebView.getSettings().setCacheMode(mWebSettings.*LOAD\_CACHE\_ELSE\_NETWORK*);

mWebView.setWebViewClient(new WebViewClient()); // 클릭시 새창 안뜨게

mWebSettings = mWebView.getSettings(); //세부 세팅 등록

mWebSettings.setJavaScriptEnabled(true); // 웹페이지 자바스클비트 허용 여부

mWebSettings.setSupportMultipleWindows(false); // 새창 띄우기 허용 여부

mWebSettings.setJavaScriptCanOpenWindowsAutomatically(false); // 자바스크립트 새창 띄우기(멀티뷰) 허용 여부

mWebSettings.setLoadWithOverviewMode(true); // 메타태그 허용 여부

mWebSettings.setUseWideViewPort(true); // 화면 사이즈 맞추기 허용 여부

mWebSettings.setSupportZoom(false); // 화면 줌 허용 여부

mWebSettings.setBuiltInZoomControls(false); // 화면 확대 축소 허용 여부

mWebSettings.setLayoutAlgorithm(WebSettings.LayoutAlgorithm.*SINGLE\_COLUMN*); // 컨텐츠 사이즈 맞추기

mWebSettings.setCacheMode(WebSettings.*LOAD\_NO\_CACHE*); // 브라우저 캐시 허용 여부

mWebSettings.setDomStorageEnabled(true); // 로컬저장소 허용 여부

mWebView.setScrollBarStyle(WebView.*SCROLLBARS\_OUTSIDE\_OVERLAY*);

mWebView.setScrollbarFadingEnabled(true);

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*) { //버전이 킷캣 이상이면

mWebView.setLayerType(View.*LAYER\_TYPE\_HARDWARE*, null); //하드웨어 가속 사용

} else {

mWebView.setLayerType(View.*LAYER\_TYPE\_SOFTWARE*, null);

}

mWebView.loadUrl("http:// ~ /"); // 웹뷰에 표시할 웹사이트 주소, 웹뷰 시작

}

@Override

public void onBackPressed() { //뒤로가기 버튼 클릭시 종료가 아닌 이전 페이지로 가도록

if(mWebView.canGoBack()){

mWebView.goBack(); //뒤로갈 페이지가 있다면 뒤로간다.

}else{

super.onBackPressed();

}

}

}

매니페스트에서도 다음 내용들을 추가

<uses-permission android:name="android.permission.INTERNET"/> 앱이 인터넷에 접속 할 수 있도록

android:hardwareAccelerated="true" 하드웨어 가속을 사용